Step 1: Load from X where:

X =

K	A	B
k1	a11	b11
k2	a21	b21

Step 2: Load from X where:

X =

K	A	B
k1	a12	b12
k3	a31	b31

Step 3: Load from Y where:

Y =

K	B	C
k1	b12	c11

Step 4: Load from Y where:

Y =

K	B	C
k1	b12	c12
k2	b22	c22

Step 5: Retract load of Step 2.

Step 6: Retract load of Step 1.

Step 7: Load from X where:

X =

K	A	B
k1	a11	b12

Figure 1

T =

K	A	B	C
k1	a11	b11	NULL
k2	a21	b21	NULL

Figure 2A

T =

K	A	B	C
k1	a12	b12	NULL
k3	a31	b31	NULL

Figure 2B

T =

K	A	B	C
k1	a12	b12	c11

Figure 2C

T =

K	A	B	C
k1	a12	b12	c12
k2	a21	b22	c22

Figure 2D

1) T after Load 1:

T =

K	A	B	C
k1	a11	b11	NULL
k2	a21	b21	NULL

2) T after Load 3:

T =

K	A	B	C
k1	a11	b12	c11

3) T after Load 4:

T =

K	A	B	C
k1	a11	b12	c12
k2	a21	b22	c22

Figure 2E

1) T after Load 3:

T =

K	A	B	C
k1	NULL	b12	c11

2) T after Load 4:

T =

K	A	B	C
k1	NULL	b12	c12
k2	NULL	b22	c22

Figure 2F

T =

K	A	B	C
k1	a11	b12	c12

Figure 2G

1) T after Load 3:

T =

K	A	B	C
k1	NULL	b12	c11

2) T after Load 4:

T =

K	A	B	C
k1	NULL	b12	c12
k2	NULL	b22	c22

3) T after Load 7

T =

K	A	B	C
k1	a11	b12	c12

Figure 3

1) T after Load 1

T =

K	A	B	C
k1	a11	b11	NULL
k2	a21	b21	NULL

2) T after Load 3:

T =

K	A	B	C
k1	a11	b12	c11

3) T after Load 4:

T =

K	A	B	C
k1	a11	b12	c12
k2	a21	b22	c22

Figure 4

1) T after Load 1:

T =

K	A	B	C
k1	a11	b11	NULL
k2	a21	b21	NULL

2) T after Load 2:

T =

K	A	B	C
k1	a12	b12	NULL
k3	a31	b31	NULL

3) T after Load 3:

T =

K	A	B	C
k1	a12	b12	c11

Figure 5

HT =

K	SR	A	SA	B	SB	C	SC	L	R
k1	A	a11	A	b11	A	NULL	N	1	NULL
k2	A	a21	A	b21	A	NULL	N	1	NULL

DT =

K	A	B	C

Figure 6A

HT =

K	SR	A	SA	B	SB	C	SC	L	R
k1	A	a11	A	b11	A	NULL	N	1	NULL
k1	A	a12	A	b12	A	NULL	N	2	NULL
k2	A	a21	A	b21	A	NULL	N	1	NULL
k2	D	NULL	D	NULL	D	NULL	D	2	NULL
k3	A	a31	A	b31	A	NULL	N	2	NULL

DT =

K	A	B	C
k2	a21	b21	NULL

Figure 6B

HT =

K	SR	A	SA	B	SB	C	SC	L	R
k1	A	a11	A	b11	A	NULL	N	1	NULL
k1	A	a12	A	b12	A	NULL	N	2	NULL
k1	A	NULL	N	NULL	C	c11	A	3	NULL
k2	A	a21	A	b21	A	NULL	N	1	NULL
k2	D	NULL	D	NULL	D	NULL	D	2	NULL
k2	D	NULL	D	NULL	D	NULL	D	3	NULL
k3	A	a31	A	b31	A	NULL	N	2	NULL
k3	D	NULL	D	NULL	D	NULL	D	3	NULL

DT =

K	A	B	C
k2	a21	b21	NULL
k3	a31	b31	NULL

Figure 6C

HT =

K	SR	A	SA	B	SB	C	SC	L	R
k1	A	a11	A	b11	A	NULL	N	1	NULL
k1	A	a12	A	b12	A	NULL	N	2	NULL
k1	A	NULL	N	NULL	C	c11	A	3	NULL
k1	A	NULL	N	NULL	C	c12	A	4	NULL
k2	A	a21	A	b21	A	NULL	N	1	NULL
k2	D	NULL	D	NULL	D	NULL	D	2	NULL
k2	D	NULL	D	NULL	D	NULL	D	3	NULL
k2	A	NULL	N	b22	A	c22	A	4	NULL
k3	A	a31	A	b31	A	NULL	N	2	NULL
k3	D	NULL	D	NULL	D	NULL	D	3	NULL
k3	D	NULL	D	NULL	D	NULL	D	4	NULL

DT =

K	A	B	C
k3	a31	b31	NULL

Figure 6D

HT =

K	SR	A	SA	B	SB	C	SC	L	R
k1	A	a11	A	b11	A	NULL	N	1	NULL
k1	A	a12	A	b12	A	NULL	N	2	5
k1	A	NULL	N	**b12**	**A**	c11	A	3	NULL
k1	A	NULL	N	NULL	C	c12	A	4	NULL
k2	A	a21	A	b21	A	NULL	N	1	NULL
k2	D	NULL	D	NULL	D	NULL	D	2	5
k2	D	NULL	D	NULL	D	NULL	D	3	NULL
k2	A	NULL	N	b22	A	c22	A	4	NULL
k3	A	a31	A	b31	A	NULL	N	2	5
k3	D	NULL	D	NULL	D	NULL	D	3	NULL
k3	D	NULL	D	NULL	D	NULL	D	4	NULL

DT =

K	A	B	C

Figure 6E

HT =

K	SR	A	SA	B	SB	C	SC	L	R
k1	A	a11	A	b11	A	NULL	N	1	6
k1	A	a12	A	b12	A	NULL	N	2	5
k1	A	NULL	N	b12	A	c11	A	3	NULL
k1	A	NULL	N	NULL	C	c12	A	4	NULL
k2	A	a21	A	b21	A	NULL	N	1	6
k2	D	NULL	D	NULL	D	NULL	D	2	5
k2	D	NULL	D	NULL	D	NULL	D	3	NULL
k2	A	NULL	N	b22	A	c22	A	4	NULL
k3	A	a31	A	b31	A	NULL	N	2	5
k3	D	NULL	D	NULL	D	NULL	D	3	NULL
k3	D	NULL	D	NULL	D	NULL	D	4	NULL

DH =

K	A	B	C

Figure 6F

HT =

K	SR	A	SA	B	SB	C	SC	L	R
k1	A	a11	A	b11	A	NULL	N	1	6
k1	A	a12	A	b12	A	NULL	N	2	5
k1	A	NULL	N	b12	A	c11	A	3	NULL
k1	A	NULL	N	NULL	C	c12	A	4	NULL
k1	A	a11	A	NULL	C	NULL	N	7	NULL
k2	A	a21	A	b21	A	NULL	N	1	6
k2	D	NULL	D	NULL	D	NULL	D	2	5
k2	D	NULL	D	NULL	D	NULL	D	3	NULL
k2	A	NULL	N	b22	A	c22	A	4	NULL
k2	D	NULL	D	NULL	D	NULL	D	7	NULL
k3	A	a31	A	b31	A	NULL	N	2	5
k3	D	NULL	D	NULL	D	NULL	D	3	NULL
k3	D	NULL	D	NULL	D	NULL	D	4	NULL

DH =

K	A	B	C
k2	NULL	b22	c22

Figure 6G

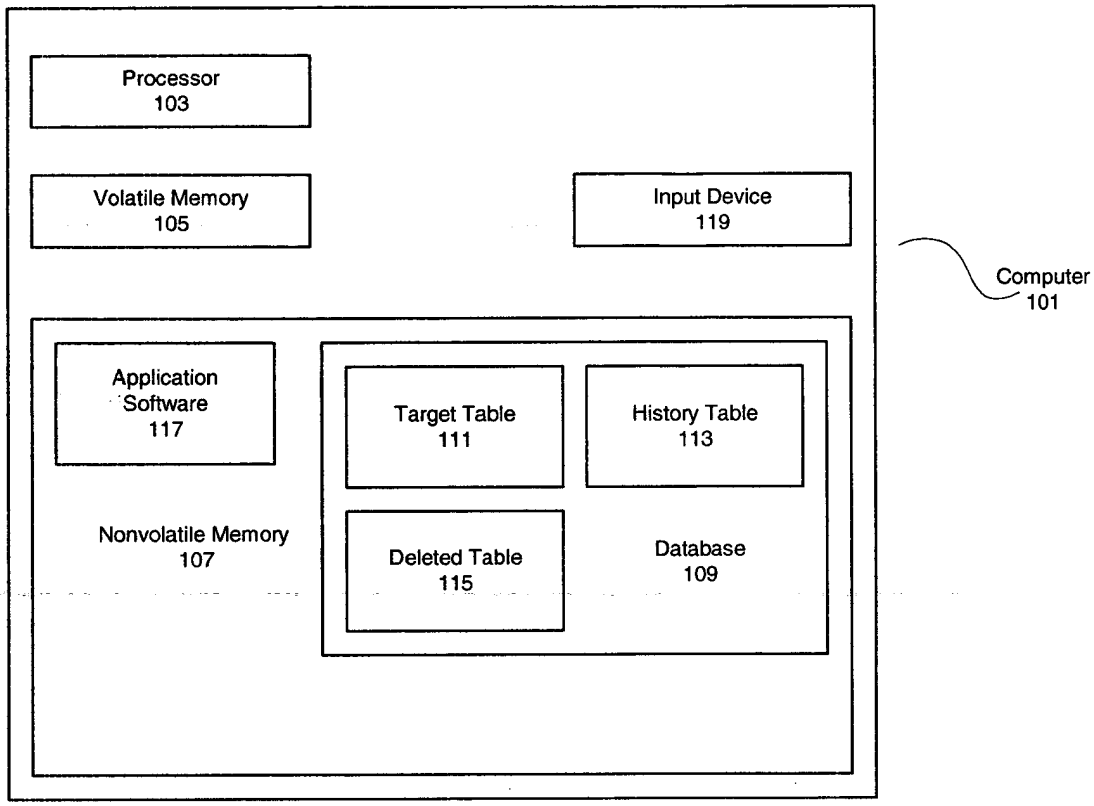

Figure 7

```
/*
The code segments below are given with respect to the following example table
declarations (written, for specificity, in Transact SQL language of Microsoft
SQL Server).
These code segments are also just one possible implementation of the invention;
other implementations may also be developed.
These code segments are also primarily given with respect to loads from table X
and retracts of loads from table X; similar code segments may be developed for
loads and retracts from table Y.
*/

CREATE TABLE T (
      K char(2) NOT NULL,
      A char(3) NULL,
      B char(3) NULL,
      C char(3) NULL
      )

CREATE TABLE DT (
      K char(2) NOT NULL,
      A char(3) NULL,
      B char(3) NULL,
      C char(3) NULL
      )

CREATE TABLE HT (
      K char(2) NOT NULL,
      SR char(1) NOT NULL,
      A char(3) NULL,
      SA char(1) NOT NULL,
      B char(3) NULL,
      SB char(1) NOT NULL,
      C char(3) NULL,
      SC char(1) NOT NULL,
      L int NOT NULL,
      R int NULL
      )

CREATE TABLE X (
      K char(2) NOT NULL,
      A char(3) NULL,
      B char(3) NULL
      )

CREATE TABLE Y (
      K char(2) NOT NULL,
      B char(3) NULL,
      C char(3) NULL
      )
```

```
-- Code segment 1A
/*
This code segment is written to process loads from table X.
*/
INSERT INTO T (K, A, B)
SELECT K, A, B
FROM X
WHERE NOT EXISTS
```

Figure 8A

```
      (SELECT 1 FROM T WHERE K=X.K)
AND NOT EXISTS
      (SELECT 1 FROM DT WHERE K=X.K)
UNION
SELECT X.K, X.A, X.B, DT.C
FROM X, DT
WHERE X.K=DT.K
AND NOT EXISTS
      (SELECT 1 FROM T WHERE X.K)
-- End of code segment 1A

-- Code segment 1B
/*
This code segment is written to process loads from table X.
*/
UPDATE T
SET A=X.A, B=X.B
FROM T, X
WHERE T.K=X.K
-- End of code segment 1B

-- Code segment 1C
/*
This code segment is written to process loads from table X.
*/
DELETE FROM T
WHERE NOT EXISTS
      (SELECT 1 FROM X WHERE K=T.K)
-- End of code segment 1C

-- Code segment 2A
/*
This code segment is written to process loads from table X.
This code segment assumes that table Inserted contains rows that were inserted
into T by code segment 1A.
Such table can be computed by application code, or may be made available
automatically by the system, as is the case in Microsoft SQL Server in an INSERT
trigger.
This code segment also assumes that integer variable @L has been declared and
initialized with the event id of the current load.
*/
INSERT INTO HT (L, SR, R, K, A, SA, B, SB, C, SC)
SELECT @L, 'A', NULL, K, A, 'A', B, 'A', NULL, 'N'
FROM Inserted
-- End of code segment 2A

-- Code segment 2B
/*
This code segment is written to process loads from table X.
This code segment assumes that tables Inserted and Deleted both contain rows
that were updated in T by code segment 1B, where table Inserted contains new
values (after the UPDATE) and table Deleted contains old values (before UPDATE).
Such tables can be computed by application code, or may be made available
automatically by the system, as is the case in Microsoft SQL Server in an UPDATE
trigger.
This code segment also assumes that integer variable @L has been declared and
initialized with the event id of the current load.
*/
```

Figure 8B

```
INSERT INTO HT (L, SR, R, K, A, SA, B, SB, C, SC)
SELECT      @L,
      'A',
      NULL,
      I.K,
      CASE WHEN D.A=I.A THEN NULL ELSE I.A END,
      CASE WHEN D.A=I.A THEN 'C' ELSE 'A' END,
      CASE WHEN D.B=I.B THEN NULL ELSE I.B END,
      CASE WHEN D.B=I.B THEN 'C' ELSE 'A' END,
      NULL,
      'N'
FROM Inserted AS I, Deleted AS D
WHERE I.K=D.K
-- End of code segment 2B

-- Code segment 2C
/*
This code segment assumes that table Deleted contains rows that were deleted
from T by code segment 1C.
Such table can be computed by application code, or may be made available
automatically by the system, as is the case in Microsoft SQL Server in a DELETE
trigger.
This code segment also assumes that integer variable @L has been declared and
initialized with the event id of the current load.
*/
INSERT INTO HT (L, SR, R, K, A, SA, B, SB, C, SC)
SELECT @L, 'D', NULL, K, NULL, 'D', NULL, 'D', NULL, 'D'
FROM Deleted
-- End of code segment 2C

-- Code segment 2D
/*
This code segment is written to process loads from table X.
This code segment assumes that variable @L has been declared and initialized
with the event id of the current load.
*/
INSERT INTO HT (L, SR, R, K, A, SA, B, SB, C, SC)
SELECT @L, 'D', NULL, K, NULL, 'D', NULL, 'D', NULL, 'D'
FROM DT
WHERE NOT EXISTS
      (SELECT 1 FROM X WHERE K=DT.K)
-- End of code segment 2D

-- Code segment 3A
/*
This code segment assumes that integer variables @L and @R have been declared
and initialized as follows: @R with the event id of the current retract, and @L
with the event id of the load being retracted.
*/
UPDATE HT
SET R=@R
WHERE L=@L
-- End of code segment 3A

-- Code segment 3B
/*
This code segment is written to process retractions of loads from table X.
```

Figure 8C

```
This code segment is written in Transact SQL language of Microsoft SQL Server,
where the copy of table HT that is updated is designated by the alias HT2 from
the first copy of table HT in the FROM clause.
Also, because attribute C is not applicable for loads from table X, it need not
be considered here.
This code segment also assumes that integer variable @L has been declared and
initialized with the event id of the load being retracted.
*/
UPDATE HT
SET A=HT1.A, SA='A'
FROM HT AS HT2, HT AS HT1
WHERE HT1.L=@L
AND HT1.SA='A'
AND HT2.K=HT1.K
AND HT2.L>HT1.L
AND HT2.R IS NULL
AND HT2.SA='C'
AND NOT EXISTS
      (SELECT 1
      FROM HT
      WHERE K=HT2.K
      AND HT1.L<L
      AND L<HT2.L
      AND R IS NULL
      AND SA IN ('A','C'))

UPDATE HT
SET B=HT1.B, SB='A'
FROM HT AS HT2, HT AS HT1
WHERE HT1.L=@L
AND HT1.SB='A'
AND HT2.K=HT1.K
AND HT2.L>HT1.L
AND HT2.R IS NULL
AND HT2.SB='C'
AND NOT EXISTS
      (SELECT 1
      FROM HT
      WHERE K=HT2.K
      AND HT1.L<L
      AND L<HT2.L
      AND R IS NULL
      AND SB IN ('A','C'))
-- End of code segment 3B

-- Code segment 4A
/*
This code segment assumes that table Deleted contains rows that were deleted
from T by code segment 1C.
Such table can be computed by application code, or may be made available
automatically by the system, as is the case in Microsoft SQL Server in a DELETE
trigger.
*/
INSERT INTO DT (K, A, B, C)
SELECT K, A, B, C
FROM Deleted
-- End of code segment 4A
```

Figure 8D

```
-- Code segment 4B
/*
This code segment assumes that table Inserted contains rows that were inserted
into T by code segment 1A.
Such table can be computed by application code, or may be made available
automatically by the system, as is the case in Microsoft SQL Server in an INSERT
trigger.
*/
DELETE FROM DT
WHERE EXISTS
      (SELECT 1 FROM Inserted WHERE K=DT.K)
-- End of code segment 4B

-- Code segment 5A
/*
This code segment assumes that integer variable @L has been declared and
initialized with the event id of the load being retracted.
*/
DELETE FROM DT
WHERE EXISTS
      (SELECT 1
      FROM HT
      WHERE L=@L
      AND K=DT.K)
-- End of code segment 5A

-- Code segment 5B
/*
This code segment assumes specifically the data types given in the example table
declarations above; it also assumes that string '***' is not a legitimate value
that can naturally appear in data.
This code segment also assumes that integer variable @L has been declared and
initialized with the event id of the load being retracted.
This code segment also assumes that the value of attribute L in table HT is
between 0 and 1147483647 inclusive.
*/
INSERT INTO DT (K, A, B, C)
SELECT HT2.K,
      CASE WHEN SubString(Max(
                  CASE WHEN HT2.SA!='A'
                        THEN NULL
                        ELSE CAST(1000000000+HT2.L AS
char(10))+IsNull(HT2.A,'***')
                  END),10,3)='***'
            THEN NULL
            ELSE SubString(Max(
                  CASE WHEN HT2.SA!='A'
                        THEN NULL
                        ELSE CAST(1000000000+HT2.L AS
char(10))+IsNull(HT2.A,'***')
                  END),10,3)
      END,
      CASE WHEN SubString(Max(
                  CASE WHEN HT2.SB!='A'
                        THEN NULL
                        ELSE CAST(1000000000+HT2.L AS
char(10))+IsNull(HT2.B,'***')
                  END),10,3)='***'
```

Figure 8E

```
            THEN NULL
            ELSE SubString(Max(
                  CASE WHEN HT2.SB!='A'
                        THEN NULL
                        ELSE CAST(1000000000+HT2.L AS
char(10))+IsNull(HT2.B,'***')
                  END),10,3)
      END,
      CASE WHEN SubString(Max(
                  CASE WHEN HT2.SC!='A'
                        THEN NULL
                        ELSE CAST(1000000000+HT2.L AS
char(10))+IsNull(HT2.C,'***')
                  END),10,3)='***'
            THEN NULL
            ELSE SubString(Max(
                  CASE WHEN HT2.SC!='A'
                        THEN NULL
                        ELSE CAST(1000000000+HT2.L AS
char(10))+IsNull(HT2.C,'***')
                  END),10,3)
      END
FROM HT AS HT2, HT AS HT1
WHERE HT1.L=@L
AND HT2.K=HT1.K
AND HT2.R IS NULL
GROUP BY HT2.K
HAVING SubString(Max(CAST(1000000000+HT2.L AS char(10))+HT2.SR),11,1)='D'
AND Sum(CASE WHEN HT2.SR='A' THEN 1 ELSE 0 END)>0
-- End of code segment 5B

-- Code segment 6A
/*
This code segment assumes that integer variable @L has been declared and
initialized with the event id of the load being retracted.
*/
DELETE FROM T
WHERE EXISTS
      (SELECT 1
      FROM HT
      WHERE L=@L
      AND K=T.K)
-- End of code segment 6A

-- Code segment 6B
/*
This code segment assumes specifically the data types given in the example table
declarations above; it also assumes that string '***' is not a legitimate value
that can naturally appear in data.
This code segment also assumes that integer variable @L has been declared and
initialized with the event id of the load being retracted.
This code segment also assumes that the value of attribute L in table HT is
between 0 and 1147483647 inclusive.
*/
INSERT INTO T (K, A, B, C)
SELECT HT2.K,
      CASE WHEN SubString(Max(
                  CASE WHEN HT2.SA!='A'
```

Figure 8F

```
                        THEN NULL
                        ELSE CAST(1000000000+HT2.L AS
char(10))+IsNull(HT2.A,'***')
                  END),10,3)='***'
            THEN NULL
            ELSE SubString(Max(
                  CASE WHEN HT2.SA!='A'
                        THEN NULL
                        ELSE CAST(1000000000+HT2.L AS
char(10))+IsNull(HT2.A,'***')
                  END),10,3)
      END,
      CASE WHEN SubString(Max(
                  CASE WHEN HT2.SB!='A'
                        THEN NULL
                        ELSE CAST(1000000000+HT2.L AS
char(10))+IsNull(HT2.B,'***')
                  END),10,3)='***'
            THEN NULL
            ELSE SubString(Max(
                  CASE WHEN HT2.SB!='A'
                        THEN NULL
                        ELSE CAST(1000000000+HT2.L AS
char(10))+IsNull(HT2.B,'***')
                  END),10,3)
      END,
      CASE WHEN SubString(Max(
                  CASE WHEN HT2.SC!='A'
                        THEN NULL
                        ELSE CAST(1000000000+HT2.L AS
char(10))+IsNull(HT2.C,'***')
                  END),10,3)='***'
            THEN NULL
            ELSE SubString(Max(
                  CASE WHEN HT2.SC!='A'
                        THEN NULL
                        ELSE CAST(1000000000+HT2.L AS
char(10))+IsNull(HT2.C,'***')
                  END),10,3)
      END
FROM HT AS HT2, HT AS HT1
WHERE HT1.L=@L
AND HT2.K=HT1.K
AND HT2.R IS NULL
GROUP BY HT2.K
HAVING SubString(Max(CAST(1000000000+HT2.L AS char(10))+HT2.SR),11,1)='A'
-- End of code segment 6B

-- Code segment 7A
/*
This code segment assumes specifically the data types given in the example table
declarations above; it also assumes that string '***' is not a legitimate value
that can naturally appear in data.
This code segment also assumes that integer variables @X and @Y have been
declared and initialized as follows: @Y with the event id of the last step of
the event sub-sequence being considered, and @X with the event id of some event
in that sub-sequence.
```

Figure 8G

```
This code segment also assumes that the value of attribute L in table HT is
between 0 and 1147483647 inclusive.
*/
DELETE FROM T
INSERT INTO T (K, A, B, C)
SELECT K,
      CASE WHEN SubString(Max(
                  CASE WHEN SA!='A'
                        THEN NULL
                        ELSE CAST(1000000000+L AS char(10))+IsNull(A,'***')
                  END),10,3)='***'
            THEN NULL
            ELSE SubString(Max(
                  CASE WHEN SA!='A'
                        THEN NULL
                        ELSE CAST(1000000000+L AS char(10))+IsNull(A,'***')
                  END),10,3)
      END,
      CASE  WHEN SubString(Max(
                  CASE WHEN SB!='A'
                        THEN NULL
                        ELSE CAST(1000000000+L AS char(10))+IsNull(B,'***')
                  END),10,3)='***'
            THEN NULL
            ELSE SubString(Max(
                  CASE WHEN SB!='A'
                        THEN NULL
                        ELSE CAST(1000000000+L AS char(10))+IsNull(B,'***')
                  END),10,3)
      END,
      CASE  WHEN SubString(Max(
                  CASE WHEN SC!='A'
                        THEN NULL
                        ELSE CAST(1000000000+L AS char(10))+IsNull(C,'***')
                  END),10,3)='***'
            THEN NULL
            ELSE SubString(Max(
                  CASE WHEN SC!='A'
                        THEN NULL
                        ELSE CAST(1000000000+L AS char(10))+IsNull(C,'***')
                  END),10,3)
      END
FROM HT
WHERE L<=IsNull(@X,L)
AND IsNull(@Y,2147483646)<IsNull(R,2147483647)
GROUP BY K
HAVING SubString(Max(CAST(1000000000+L AS char(10))+SR),11,1)='A'
-- End of code segment 7A
```

Figure 8H